1. Model Approach

I chose the XGBoost model for this prediction task this week. XGBoost is renowned for its efficient and high-performance capabilities. Its strength lies in the ensemble technique of gradient boosting, which systematically corrects the errors of weak learners in successive iterations. Given the intricate nature of datasets where interrelationships between features may not be straightforward, the choice of XGBoost makes sense. It's especially adept at teasing out underlying patterns and relationships, offering superior predictive prowess.

This model's complexity reflected that while XGBoost offers a plethora of advantages, it's more sophisticated compared to simpler models like Logistic Regression. XGBoost constructs a series of decision trees and judiciously amalgamates their outputs. This deep ensemble approach undeniably captures more nuances of the data but at the cost of added computational intensity. Furthermore, fine-tuning its hyperparameters, given its complexity, requires careful consideration to prevent overfitting and ensure model generalization.

2. Hyperparameters evaluated

'n_estimators': This determines the number of trees to be employed. While a greater number can unearth more intricate patterns, it also risks overfitting.

'max_depth': This sets a cap on how deep each tree can grow. A deeper tree can discern more details but can also lead to overfitting.

'gamma': Acts as a regularization term. It stipulates the minimum loss reduction required to make a further partition.

'subsample & colsample_bytree': Both are techniques to prevent overfitting by randomizing the data and features used in each iteration.

These hyperparameters were chosen for evaluation because of their pivotal role in model performance and their ability to act as countermeasures against overfitting.

3. Model Performance Metrics

The metrics of choice are accuracy and AUC-ROC:

a. Accuracy: This is the most straightforward metric, signifying the proportion of correct predictions. It's particularly apt for balanced datasets.

b. AUC-ROC: This metric is instrumental in gauging the model's capacity to differentiate between the positive and negative classes. Especially for imbalanced datasets, AUC-ROC is a more robust metric than mere accuracy. These metrics offer a comprehensive assessment of how well the XGBoost performs in the given prediction task.

4. Metrics Calculation

I use the ‘cross_val_score’ function to calculate the accuracy of the training data. The best cross-validation score is 0.8805355790273499. Additionally, with ‘roc_curve’, I computed the AUC-ROC for the validation dataset.

5. Analysis and Metrics Discussion

Model Variation	Train AUC	Validation AUC
Base Model	0.9653	0.9429
Tree Hyperparameters	0.9550	0.9409
Boosting Hyperparameters	0.9302	0.9268

a. Base Model:

This is the standard XGBoost classifier without any hyperparameter tuning. It provides a solid baseline to compare other models against. Given its high AUC score, it's evident that the features and the model architecture are inherently powerful.

b. Tree Hyperparameters:

This variation likely focuses on parameters associated with individual trees in the ensemble such as ‘max_depth’, ‘min_child_weight’, and ‘gamma’. The aim is generally to prevent overfitting and create more generalized models.

c. Boosting Hyperparameters:

This model probably focuses on tuning parameters that guide the boosting process, such as ‘learning_rate’, ‘n_estimators’, ‘subsample’, and ‘colsample_bytree’. Boosting hyperparameters helps the model train sequentially to correct the mistakes of previous trees.

d. Analysis:

The base model has the highest AUC for both training and validation, suggesting that it might be the most robust model. The Tree Hyperparameters model has a slightly reduced training AUC, indicating that it might be less prone to overfitting compared to the base model. However, its validation AUC is marginally lower than the base model. The Boosting Hyperparameters model shows a significant decrease in both training and validation AUC, suggesting that the specific hyperparameters tuned in this model may not be optimal for this dataset. When looking at the validation, the base model performs best, followed closely by the tree hyperparameters model. The boosting hyperparameters model lags.